PAŃSTWOWA WYŻSZA SZKOŁA ZAWODOWA
W NOWYM SĄCZU

Instytut Techniczny
Informatyka Stosowana

DOKUMENTACJA PROJEKTOWA

PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

Aplikacja Zdrowotna

Autorzy:
Piotr Szczepanek
Arkadiusz Homoncik

Prowadzący:
mgr inż. Dawid Kotlarski

Nowy Sącz 2021

Spis treści

1. Ogólne określenie wymagań **3**

1.1. Wymagania zleceniodawcy dotyczące programu 3

2. Określenie wymagań szczegółowych **6**

2.1. Specyfikacja aplikacji 6

3. Projektowanie **8**

3.1. Wygląd aplikacji 8
3.2. Podstrony aplikacji 8
3.3. Uruchomienie aplikacji na smartphone'ie 9
3.4. Git 9
3.5. Wykrywanie lokalizacji 9
3.6. Odcisk Palca 10
3.7. Twoje postępy 11
3.8. Ostateczny wygląd aplikacji 11

4. Implementacja **15**

4.1. Przełączanie pomiędzy stronami 15
4.2. Lokalizacja 16
4.3. Zabezpieczenia biometryczne 17
4.4. Krokomierz 19
4.5. Zapisywanie postępów 21
4.6. Barometr 24

5. Testowanie **28**

6. Podręcznik użytkownika **29**

Literatura **30**

Spis rysunków **31**

Spis tabel **32**

1. Ogólne określenie wymagań

1.1. Wymagania zleceniodawcy dotyczące programu

Zamówienie

1. Rodzaj aplikacji:
Dzień dobry, jestem przedstawicielem firmy Żyj Zdrowo. Nasza firma od wielu lat zajmuje się świadczeniem wysokiej jakości usług, których celem jest dbanie o dobry stan zdrowia fizycznego jak i psychicznego naszych klientów. Nasza firma stale się rozwija. Mamy kilka siedzib rozmieszczonych na południu naszego kraju, prowadzimy siłownie, kluby fitness, jak również ostatnio weszliśmy ze świadczeniem naszych usług na rynki internetowe z nadzieją na dotarcie do większej grupy odbiorców, którzy potrzebują trenerów personalnych. Ostatnio mierzymy się z problemem zachęcenia nowych klientów do siebie. Dlatego zwracamy się do Państwa z prośbą stworzenia aplikacji, która w łatwiejszy sposób pomogła by nam dotrzeć do naszych klientów oraz byłaby czymś w rodzaju reklamy zwiększającej naszą rozpoznawalność.

2. Charakterystyka aplikacji:
Potrzebujemy aplikacji zdrowotnej charakteryzującą się tym że: - będzie posiadała łatwy i przystępny interfejs dla użytkownika
- będzie zawierała logo naszej firmy
- zmierzy osiągi użytkownika
- będzie dostępna dla urządzeń z systemem android
- nasza aplikacja powinna być dostępna zarówna na nowoczesnych urządzeniach, jak i również urządzeniach starszego typu (ma być uruchamiana na urządzeniach ze słabszymi komponentami)
- aplikacja będzie mierzyć prędkość użytkownika na podstawie systemu GPS
- ma mierzyć liczbę kroków
- będzie pokazywała aktualne położenie użytkownika (w postaci np. Miejscowości)
- aplikacja będzie zliczała liczbę kroków użytkownika (będzie posiadać krokomierz)
- aplikacja będzie zapisywała postęp osiągnięć użytkowników
- użytkownik podczas aktywności będzie miał możliwość sprawdzenia swoich czynności życiowych poprzez urządzenia takiego typu jak żyroskop.
- użytkownik będzie mógł założyć zabezpieczenie biometryczne na swoją aplikację np. poprzez odcisk palców
- postępy użytkownika będą widoczne na wykresach

- aplikacja będzie miała możliwość generowania raportów tygodniowych odnośnie postępów użytkownika
- użytkownik będzie w stanie za pomocą aplikacji sprawdzić swój miesięczny postęp z tygodnia na tydzień
- aplikacja będzie w stanie liczyć spalone kalorie użytkownika
- aplikacja będzie proponować użytkownikowi na podstawie jego aktywności fizycznej jedną z naszych diet
- Nasza aplikacja będzie umożliwiała wypisywanie powiadomień odnośnie osiągniętych celów na urządzeniach typu smartwatch, smartband
- porównuje osiągi z innymi użytkownikami
- ma możliwość pokazywania innych użytkowników w terenie
- wysyła do innych użytkowników informacje o naszej aktywności
- możliwa praca na komputerach
- zapewnienie działania dla systemów iOS

3. Wygląd aplikacji:

Chcemy aby nasza aplikacja posiadała nowoczesny wygląd. Naszym założeniem jest, aby jej interfejs posiadał zaawansowaną szatę graficzną, aby korzystanie z niej było przyjazne dla użytkowników, np. poprzez ciekawą kolorystykę. Mile widziane byłoby również, gdyby aplikacja posiadała nowoczesne przejścia wizualne. Użytkownik podczas wejścia do aplikacji powinien bez problemowo wiedzieć, w której zakładce może się spodziewać poszczególnych opcji. W pierwszej zakładce powinny znajdować się przyciski od poszczególnych opcji.

4. Grupa docelowa:

Byłoby nam niezmiernie miło, gdyby nasza aplikacja była skierowana do szerokiej grupy odbiorców. Chcielibyśmy, aby mogły z niej korzystać nie tylko osoby młode, ale również, aby była dostosowana dla osób starszych. Poprzez zaawansowane opcje aplikacji oraz jej rozbudowany interfejs, chcielibyśmy dotrzeć do nie tylko do naszych stałych klientów, ale również dzięki tej aplikacji pozyskać nowych.

5. Koszt aplikacji, terminy i raporty:

Niestety jako przedstawiciel naszej firmy nie posiadam fachowej wiedzy ile może kosztować zaproponowany przeze mnie program. Dobrze by było, żeby aplikacja nie wymagała dużych nakładów pieniężnych. Dlatego chcielibyśmy prosić o wycenę kosztów oraz wybranie tych funkcji aplikacji, z których ze względu na koszty lub problemy w zaimplementowaniu można by było zrezygnować. Liczymy również na

to, że będą państwo informować nas o dokonanych w naszej aplikacji postępach poprzez wysyłanie co tygodniowych raportów.

2. Określenie wymagań szczegółowych

2.1. Specyfikacja aplikacji

Specyfikacja aplikacji

1. Ogólny opis aplikacji (wygląd)

Podejmujemy się stworzenia aplikacji zdrowotnej o nazwie „Żyj zdrowo". Aplikacja ta ma być stworzona z myślą o użytkownikach prowadzących aktywny i zdrowy tryb życia. Aplikacja ma być tworzona z myślą o prostym i intuicyjnym użytkowaniu. Wstępny zarys wyglądu aplikacji przewiduje stworzone przez nas logo znajdujące się na górze ekranu, pod którym ma się znajdować blok z główną funkcją aplikacji informującej o liczbie przebytych kroków. Pod nim znajdować się będą inne bloki z pozostałymi funkcjami aplikacji. Bloki będą ułożone w formie listy selektywnej, z której użytkownik będzie wybierać interesujące go funkcje. Kolorystyka i wygląd będzie stworzona z myślą o wygodzie użytkowania.

2. Charakterystyka aplikacji

Wygląd zgodny z opisem aplikacji z możliwością zmiany wstępnego zarysu. Aplikacja będzie zgodna z systemem android z możliwością uruchomienia na wszystkich urządzeniach z tym systemem. Wszystkie podstawowe funkcje, do których aplikacja zostanie stworzona (takie jak: krokomierz, geolokalizacja, pulsometr, zabezpieczenie biometryczne), będą działać prawidłowo. Zgodnie z założeniami aplikacji będzie ona zapisywała postęp osiągnięć naszych użytkowników. Opcjonalnymi funkcjami będą: informacje o postępach użytkownika za pomocą wykresów. Zaimplementowany zostanie również prędkościomierz.

3. Trudne w implementacji

Po przeanalizowaniu listy elementów aplikacji z zamówienia rezygnujemy z trudnych w implementacji funkcji takich jak: a) liczenie spalonych kalorii, ponieważ brak jest możliwości sprawdzenia poprawności działania

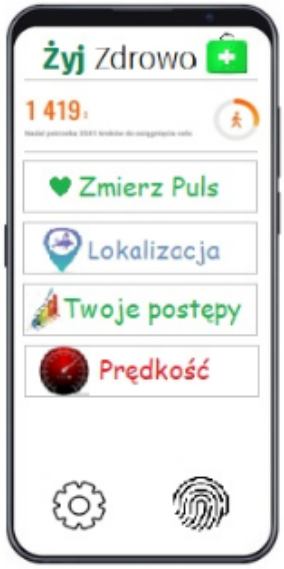
b) diety – wymaga podłączenia się z bazą danych klienta, co zwiększyło by koszty produkcji oraz skomplikowało proces tworzenia aplikacji

c) możliwość połączenia aplikacji ze smartbandem i smartwatchem – nasza firma nie zajmuje się tego typu rozwiązaniami

d) porównywanie osiągów z innymi użytkownikami może być trudne i czasochłonne w implementacji, ponieważ wymagałoby to stworzenia przez nas bazy danych, zawierającej informacje oraz dane pozostałych użytkowników (zwiększyłoby to koszty

produkcji aplikacji)
e) użytkowanie na komputerach nie będzie możliwe – wymagałoby to stworzenia oddzielnej aplikacji dla komputerów
f) brak możliwości sprawdzenia działania aplikacji na urządzeniach z systemem iOS w fazie testów

4. Koszt aplikacji i raporty:
W cotygodniowych raportach będziemy zamieszczać informacje o postępach jak o tym czego nie dało się zrealizować. Koszt aplikacji zostanie zmniejszony ze względu na rezygnacje z w trudnych w implementacji funkcji.



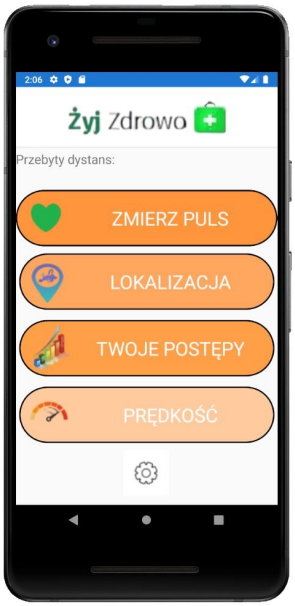
Rys. 2.1. Grafika Koncepcyjna

3. Projektowanie

Aplikacja jest tworzona przy pomocy środowiska Xamarin.

3.1. Wygląd aplikacji

Menu główne aplikacja zawiera prosty i intuicyjny interfejs, w którym rozmieszczone są przyciski w postaci listy selektywnej, z której można wybrać jedną z funkcji aplikacji. Wygląd strony głównej i przyciski zawierają pomarańczową szatę graficzną. Przyciski zawierają grafikę, która symbolizuje właściwość wybieranej funkcji. Na górze ekranu zamieszczone zostało logo aplikacji. Rys. 3.1. przedstawia początkowy wygląd aplikacji.



Rys. 3.1. Wygląd aplikacji

3.2. Podstrony aplikacji

Spersonalizowaliśmy przyciski, odpowiedzialne kolejno za : pomiar pulsu, geolokalizację użytkownika, jego postępy oraz prędkość. Po naciśnięciu przycisku na stronie głównej wykonuje się akcja, która pozwala na przejście do stron z określnymi funkcjami. Nowe strony zostały przygotowane na wdrożenie poszczególnych usług.

3.3. Uruchomienie aplikacji na smartphone'ie

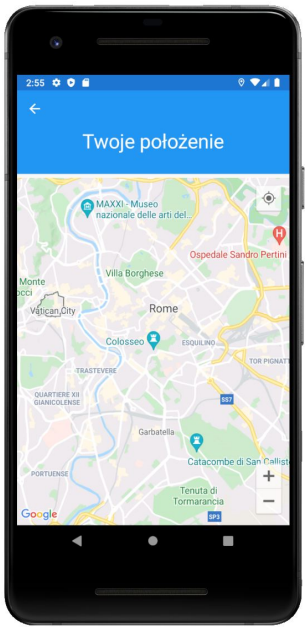
Podczas tworzenia aplikacji sprawdziliśmy działanie naszej aplikacji przez pobranie jej na naszego smartphone'a. W pobraniu aplikacji wykorzystaliśmy tutorial[1].

3.4. Git

Napotkaliśmy problem z gitem polegający na braku możliwości dodania projektu powyżej 100MB. Aby dodać taki projekt konieczne jest skorzystanie z GIT LFS[2].

3.5. Wykrywanie lokalizacji

Poprzez przycisk LOKALIZACJA użytkownik jest w stanie przejść do widoku map i zobaczyć swoją aktualną lokalizację wraz z otoczeniem – miejscami, które znajdują się w pobliżu. W tym celu skorzystaliśmy z API firmy Google oraz poradnika na Youtube'ie[3]. Konieczne było doinstalowanie pakietu: Xamarin.Forms.Apps. Rys. 3.2. ukazuje wygląd opcji Twoje położenie.



Rys. 3.2. wygląd po implementacji

[1]tutorial jak wygenerować Xamarin Android APK[**tutorial1**].
[2]Git Large File Storage - How to Work with Big Files[**tutorial2**].
[3]Xamarin.Forms.Apps tutorial[**Xamarin.Forms.Apps**].

3.6. Odcisk Palca

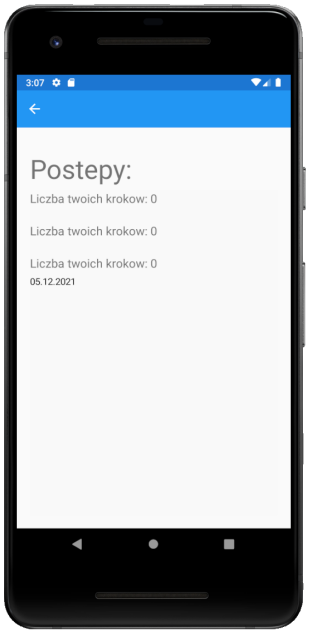
Dodaliśmy również ustawienia biometryczne w postaci odcisku palca. W celu dodania możliwości odblokowania za pomocą odcisku palca, konieczne było doinstalowanie pakietu: Plugin.Fingerprint. [4][5][6][7] Wygląd strony Ustawienia przedstawia Rys. 3.3.

Rys. 3.3. wygląd po implementacji ustawienia

[4]Biometric Authentication With Fingerprint[**Authentication1**].
[5]Authentication in Xamarin Forms[**Authentication2**].
[6]Secure Your Xamarin App with Fingerprint or Face Recognition[**Authentication3**].
[7]Wprowadzenie z uwierzytelnianiem odciskiem palca[**Authentication4**].

3.7. Twoje postępy

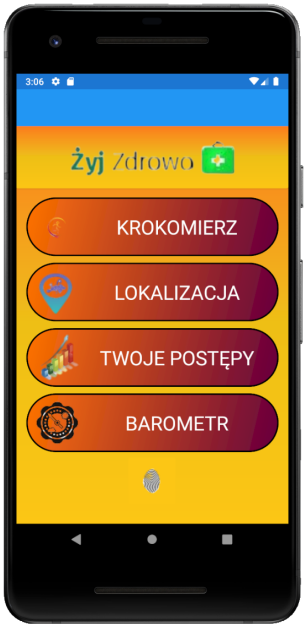
Kroki użytkownika zapisują się od teraz w jego postępach. Dokonaliśmy tego poprzez podłączenie bazy danych, która zapisuje kroki użytkownika aplikacji. Musi on tylko podać datę, kiedy chce policzyć swoje kroki oraz potwierdzić to przyciskiem, który służy za zapisanie tej daty do bazy danych.



Rys. 3.4. Twoje postępy

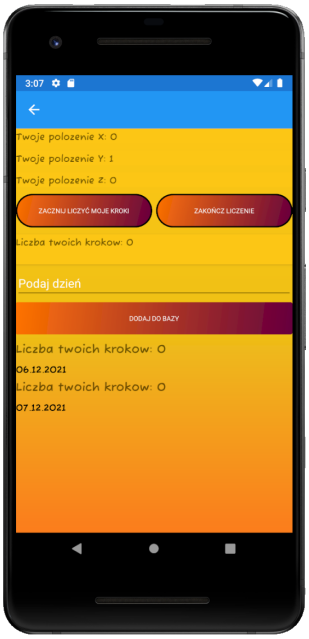
3.8. Ostateczny wygląd aplikacji

Po zaimplementowaniu wszystkich funkcji w naszej aplikacji zajeliśmy się zmianą i ulepszeniem wyglądu całej aplikacji, aby była ona bardziej przystępna ze względów wizualnych oraz spełniała wcześniej założone oczekiwania. Ostateczny wygląd aplikacji zawiera: gradientowe przyciski, zmienione tło każdej ze stron oraz zmieniony rodzaj czcionki. Dzięki tym zmianom aplikacja jest bardziej przejrzysta i czytelna dla użytkownika. Wygląd aplikacji przedstawiają poniższe grafiki. Wszystkie zmiany zostały zaimplementowane w plikach XAML. Poszczególne rysunki przedsatiwają: Rys. 3.5. Strona główna, Rys. 3.6. Barometr, Rys. 3.7. krokomierz, Rys. 3.8. Postępym, Rys. 3.9. Ustawienia.



Rys. 3.5. Strona główna - wygląd

Rys. 3.6. Barometr - wygląd



Rys. 3.7. Krokomierz - wygląd

Rys. 3.8. Postępy - wygląd

Rys. 3.9. Ustawienia - wygląd

Przykład implementacji przycisku z pliku MainPage.xaml przedstawia Rys. 3.10.

```
<Button x:Name="krokomierz" Clicked="krokomierz_Clicked" Text="krokomierz"  ImageSource="krokomierz.png"
        Grid.Column="0" Grid.Row="0"  FontSize="30" TextColor="White"
        FontFamily="Comic Sans MS" BorderRadius="10"
        Margin="15,2,15,2"  BorderWidth="2" BorderColor="Black" CornerRadius="50">
    <Button.Background>
        <LinearGradientBrush>
            <GradientStop Color="#FF7308" Offset="0.0"/>
            <GradientStop Color="#66023D" Offset="1.0"/>
        </LinearGradientBrush>
    </Button.Background>
</Button>
```

Rys. 3.10. Implementacja gradientowego przycisku

4. Implementacja

4.1. Przełączanie pomiędzy stronami

Wdrożyliśmy akcje przycisków z mainPage aplikacji, która umożliwia przełączenie się pomiędzy poszczególnymi funkcjami aplikacji. Dodaliśmy w tym celu nowe pliki xaml oraz zaimplementowaliśmy metody w plikach cs odpowiedzialne za poszczególne akcje.[8]. Rys. 4.1. przedstawia fragment kodu w pliku MainPage.xaml odpowiedzialnego za wygląd strony głównej aplikacji.

```
<Frame>
    <Image Source="logo.jpg">

    </Image>

</Frame>
<ScrollView></ScrollView>
    <Label Text="Przebyty dystans: " HeightRequest="50"
           FontSize="20" FontFamily="Comic Sans MS"  />
    <Grid RowSpacing="10">
<Grid.ColumnDefinitions>
    <ColumnDefinition Width="*" />

</Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="*" />
        <RowDefinition Height="*" />
        <RowDefinition Height="*" />
        <RowDefinition Height="*" />
    </Grid.RowDefinitions>


</Grid>

<Button x:Name="krokomierz" Clicked="krokomierz_Clicked" Text="krokomierz"
    Grid.Column="0" Grid.Row="0" FontSize="30" TextColor="White"
    FontFamily="Comic Sans MS" BorderRadius="10" BackgroundColor="#ff953e"
    BorderWidth="2" BorderColor="Black" CornerRadius="50"/>
```

Rys. 4.1. Kod wyglądu strony głównej aplikacji

W pliku MainPage.xaml.cs zaimplementowane zostały funkcje odpowiadające za zmianę strony w aplikacj (rys. 4.2. przedstawia fragment kodu MainPage.xaml.cs).

[8]Przełączanie stron aplikacji[**tutorial3**]

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;

[assembly: ExportFont("ComicSansMS3.ttf")]
[assembly: ExportFont("FA5Regular.otf", Alias = "FontAwesome")]

namespace App3
{
    Odwołania: 5
    public partial class MainPage : ContentPage
    {
        1 odwołanie
        public MainPage()
        {
            InitializeComponent();
        }

        Odwołania: 0
        private void Postepy_Clicked(object sender, EventArgs e)
        {
            Navigation.PushAsync(new Postepy());
        }
```

Rys. 4.2. Kod w pliku MainPage.xaml.cs

4.2. Lokalizacja

Plik AndroidManifest.xaml jest odpowiedzialny w naszej aplikacji za umożliwienie dostępu użytkownikowi aplikacji do różnego rodzaju czujników i funkcji dostępnych w telefonie. Na Rys. 4.3. zostało użyte pozwolenie, które umożliwia dostęp do lokalizacji.

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.USE_BIOMETRIC" />
<uses-feature android:name="android.hardware.sensor.accelerometer" android:required="true" />
```

Rys. 4.3. AndroidManifest.xaml

W pliku MainActivity.cs (Rys. 4.4.) zaimplementowaliśmy lokalizację. Skorzystaliśmy z API firmy Google, które wkleiliśmy w linijce META-DATA, a dokładniej w ANDROID:VALUE.

```
protected override void OnCreate(Bundle savedInstanceState)
{
    CrossFingerprint.SetCurrentActivityResolver(() => this);

    TabLayoutResource = Resource.Layout.Tabbar;
    ToolbarResource = Resource.Layout.Toolbar;

    base.OnCreate(savedInstanceState);

    Xamarin.Essentials.Platform.Init(this, savedInstanceState);
    global::Xamarin.Forms.Forms.Init(this, savedInstanceState);
    Xamarin.FormsMaps.Init(this, savedInstanceState);
    LoadApplication(new App());
}
```

Rys. 4.4. MainActivity.cs

4.3. Zabezpieczenia biometryczne

```
private async void AuthButton_Clicked(object sender, EventArgs e)
{
    bool isFingerprintAvailable = await CrossFingerprint.Current.IsAvailableAsync(false);
    if (!isFingerprintAvailable)
    {
        await DisplayAlert("Błąd",
            "Ustawienia biometryczne nie są dostępne lub nie zostały jeszcze skonfigurowane ", "OK");
        return;
    }

    AuthenticationRequestConfiguration conf =
        new AuthenticationRequestConfiguration("Zabezpieczenia",
        "Zabezpieczenie do Twoich danych");

    var authResult = await CrossFingerprint.Current.AuthenticateAsync(conf);
    if (authResult.Authenticated)
    {
        //Success
        await DisplayAlert("Sukces", "Tożsamość potwierdzona", "OK");
    }
    else
    {
        await DisplayAlert("Błąd", "Nie można zidentyfikować odcisku palca", "OK");
    }
}
```

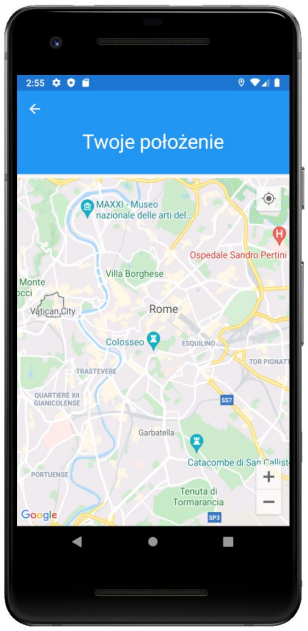
Rys. 4.5. Ustawienia.xaml.cs

W powyższych plikach zapisaliśmy kod w języku C#, odpowiedzialny za implementację funkcji lokalizacji oraz odcisku palca (Rys. 4.5.). Jak widzimy na screen'ie, użytkownik na samym początku musi przyznać aplikacji dostęp do swojej lokalizacji. Bez tego funkcja nie zadziała. Podobnie jest z odciskiem palca. Zabezpieczenie to, musi najpierw zostać skonfigurowane na Naszym urządzeniu. Na tej podstawie, aplikacja sprawdza, czy odczytany odcisk palca jest prawidłowy, czy też nie.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:maps="clr-namespace:Xamarin.Forms.Maps;assembly=Xamarin.Forms.Maps"
             x:Class="App3.Lokalizacja">
    <ContentPage.Content>
        <StackLayout>
            <Frame BackgroundColor = "#2196F3" Padding="0,0,0,30" CornerRadius="0">
                <Label Text="Twoje położenie" FontSize="30" HorizontalTextAlignment="Center" TextColor="White"/>
            </Frame>

            <maps:Map IsShowingUser="True"/>
            </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

Rys. 4.6. lokalizacja.xaml

Na powyższym zdjęciu widzimy plik xaml ukazany na rysunku Rys. 4.6. odpowiedzialny za wygląd, który ukaże Nam się po naciśnięciu przycisku LOKALIZACJA. Klasa Map definiuje właściwość typu bool - IsShowingUser. Domyślnie właściwość ta oznacza, że jeśli wartość jest ustawiona na false, to nie pokazuje bieżącej lokalizacji użytkownika. Gdy ta właściwość jest ustawiona na true, mapa pokazuje bieżącą lokalizację użytkownika. W powyższym przykładzie pokazano ustawianie tej właściwości. Na Rys. 4.7. ukazany jest wygląd funkcji pokazywania lokalizacji na emulatorze.



Rys. 4.7. lokalizacja

Na powyższym zdjęciu widzimy plik xaml (Rys. 4.8.) odpowiedzialny za wygląd, który ukaże Nam się po naciśnięciu przycisku ustawień, a dokładniej opcji biometrycznych. Ustawiliśmy tu takie rzeczy jak czcionkę, rozmiar tekstu, kolor przycisku, jego zaokrąglenia, kolor tekstu itd. Rys. 4.9. przedstawia wygląd opcji ustawienia.

```
<ContentPage.Content>
    <StackLayout>
        <Label Text="Ustawienia" FontSize="80"/>
        <Button Text="Odcisk Palca" x:Name="AuthButton"
            Clicked="AuthButton_Clicked"
            HorizontalOptions="Center"
            VerticalOptions="CenterAndExpand"
            Grid.Column="0" Grid.Row="0" FontSize="30" TextColor="White"
            FontFamily="Comic Sans MS" BorderRadius="10" BackgroundColor="#ff953e"
            BorderWidth="2" BorderColor="Black" CornerRadius="50"/>
    </StackLayout>
</ContentPage.Content>
```

Rys. 4.8. ustawienia.xaml

Rys. 4.9. Ustawienia

4.4. Krokomierz

System krokomierza w swoim działaniu wykorzystuje akcelerometr, żeby z niego skorzystać wykorzystaliśmy bibliotekę Xamarin.Esentials. W implementacji pobierane zostają trzy wartości (x, y, z) wysyłane przez akcelerometr. Funkcja odpowiedzialna za pobieranie wartości z akcelerometru ukazana została na Rys. 4.10. Pobieranie wartości przez akcelerometr. Są to wartości, które określają dokładne przyspieszenie osiowe smartfona w przestrzeni. Do obliczania kroków wykorzystaliśmy algorytm, obliczający wektor przyspieszenia, który jest pierwiastkiem kwadratów $(x^2+y^2+z^2)$ z wartości osi X, Y i Z. Następnie oblicza średnie wartości przyspieszenia przez pobranie wartości poprzedniej wektora przyspieszenia i dodanie wartości aktualnego wektora. Następnie porównuje średnie wartości przyspieszenia z wartościami progo-

wymi, aby zliczyć liczbę kroków (wartość progowa jest z góry ustalana). Dodatkowo pomiędzy pobierane wartości z akcelerometru dodane jest opóźnienie wynoszące w tym przypadku 500ms. Jeżeli wektor przyspieszenia przekracza wartość progową, to zwiększa liczbę kroków; w przeciwnym razie odrzuca nieprawidłowe wibracje[9]. Implementacja działania krokomierza została przedstawiona na Rys. 4.11. .

```
private void Start_Clicked(object sender, EventArgs e)
{
    if (Accelerometer.IsMonitoring)
        return;
    Accelerometer.ReadingChanged += Accelerometer_ReadingChanged;
    Accelerometer.Start(SensorSpeed.UI);



}
```

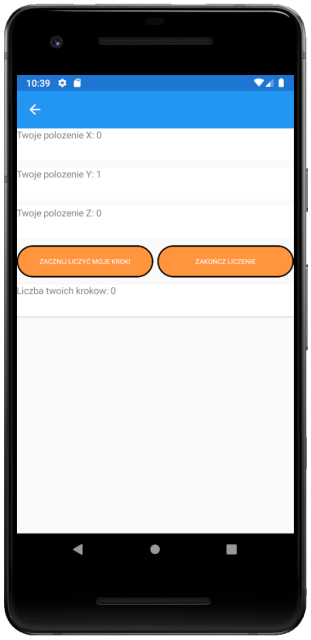
Rys. 4.10. Pobieranie wartości przez akcelerometr

```
//opoznienie pomiedzy krokami
await Task.Delay(500);

double x = e.Reading.Acceleration.X;
double y = e.Reading.Acceleration.Y;
double z = e.Reading.Acceleration.Z;

//opoznienie pomiedzy krokami
await Task.Delay(500);

double wektorPrzyspieszenia = Math.Sqrt((x * x) + (y * y) + (z * z));
double delta = (wektorPrzyspieszenia + wartoscPoprzednia) / 2;
wartoscPoprzednia = wektorPrzyspieszenia;

//stepCount++;

if (delta > 1.5)
{
    stepCount++;
}


LabelX.Text = "Twoje polozenie X: " + e.Reading.Acceleration.X.ToString();
LabelY.Text = "Twoje polozenie Y: " + e.Reading.Acceleration.Y.ToString();
LabelZ.Text = "Twoje polozenie Z: " + e.Reading.Acceleration.Z.ToString();
LabelWynik.Text = "Liczba twoich krokow: " + stepCount.ToString();
```

Rys. 4.11. Implementacja krokomierza

W celach tesowych pozostawiliśmy informacje o położeniu każdej osi pobranych z akcelerometru. Rys. 4.12. przedstawia wygląd aplikacji krokomierz.

[9]implementacja krokomierza [**implementacja krokomierza w arduino**].



Rys. 4.12. Krokomierz

4.5. Zapisywanie postępów

Do zapisywania wyników na stronie aplikacji Twoje postępy należało stowrzyć model bzy danych. W tym celu należało zainstalować pakiet NuGet sqlite-net-pcl. W aplikacji dodaliśmy nową klasę Wynik.cs. Po dodaniu tej klasy należało w pliku Database.cs (Rys. 4.14) zaimplementować połączenie z bazą do której będą zapisywane nasze wyniki i postępy z krokomierza, jak również możliwość zapisu i odczytu z tej bazy. Rys. 4.13. przedstawia model bazy danych.

```
namespace App3
{
    Odwołania: 5
    public class Wynik
    {
        [PrimaryKey, AutoIncrement]
        Odwołania: 0
        public int Id { get; set; }
        1 odwołanie
        public string Name { get; set; }
        1 odwołanie
        public string Dzien { get; set; }
    }
}
```

Rys. 4.13. Model bazy danych

```
public class Database
{
    private readonly SQLiteAsyncConnection _database;

    1 odwołanie
    public Database(string dbPath)
    {
        _database = new SQLiteAsyncConnection(dbPath);
        _database.CreateTableAsync<Wynik>();
    }

    Odwołania: 3
    public Task<List<Wynik>> GetPeopleAsync()
    {
        return _database.Table<Wynik>().ToListAsync();
    }

    1 odwołanie
    public Task<int> SavePersonAsync(Wynik person)
    {
        return _database.InsertAsync(person);
    }
}
```

Rys. 4.14. Połączenie z bazą danych

W pliku Postępy.xaml (Rys. 4.15.) utworzyliśmy interfejs, który zapisuje bezpośrednio poszczególne wyniki z bazy danych do strony aplikacji (w tym celu skorzystaliśmy ze znacznika collectionView).

```
<CollectionView x:Name="collectionView">
    <CollectionView.ItemTemplate>
        <DataTemplate>
            <StackLayout>
                <Label Text="{Binding Name}"
                  FontSize="Medium" />
                <Label Text="{Binding Dzien}"
                  TextColor="Black"
                  FontSize="Small" />
            </StackLayout>
        </DataTemplate>
    </CollectionView.ItemTemplate>
</CollectionView>
```

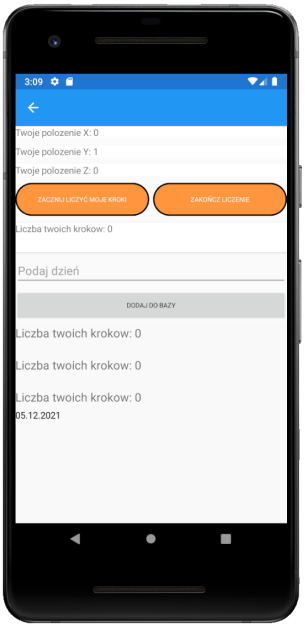
Rys. 4.15. Postępy.xaml

Do pliku krokomierz.xaml.cs (Rys. 4.16.) dodaliśmy akcje do przycisku, która zapisuje wszystkie wyniki w bazie dzięki czemu możliwe jest zapisanie ich w zakładce Twoje postępy. Również tam znajduje się funkcja, która umożliwia dodanie nowych wyników po ich zapisie. Ta sama funkcja znajduje się w pliku postepy.xaml.cs.

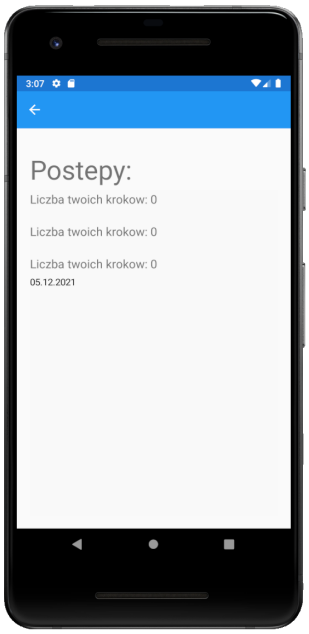
```
protected override async void OnAppearing()
{
    base.OnAppearing();
    collectionView.ItemsSource = await App.Database.GetPeopleAsync();
}

Odwołania: 0
async void OnButtonClicked(object sender, EventArgs e)
{
    if (!string.IsNullOrWhiteSpace(LabelWynik.Text))
    {
        await App.Database.SavePersonAsync(new Wynik
        {
            Name = LabelWynik.Text,
            Dzien = dateEntry.Text

        });

        LabelWynik.Text = string.Empty;


        collectionView.ItemsSource = await App.Database.GetPeopleAsync();
    }
}
```

Rys. 4.16. Krokomierz.xaml.cs

Poniżej znajdują sie strony aplikacji, na których zapisane są osiągnięte wyniki z bazy danych w Krokomierz (Rys. 4.17) i Postępy (Rys. 4.18.).



Rys. 4.17. Krokomierz



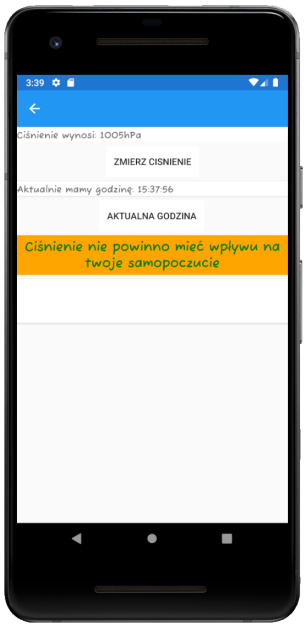
Rys. 4.18. Twoje postępy

4.6. Barometr

Aplikacja ma zaimplementowaną funkcję barometru. Dzięki tej funkcji użytkownik jest w stanie sprawdzić, jakie aktualnie jest ciśnienie atmosferyczne oraz wyświetlić aktualną godzinę pomiaru. Godzina jest pobierana automaycznie z telefonu[10][11]. Dane te są wykorzystywane dalej w Naszej aplikacji. Ciśnienie atmosferyczne ma ogromną rolę i może wywierać duży wpływ na Nasze samopoczucie[12][13]. Przykładowo jeżeli ciśnienie wynosi powyżej 1000 hPa, użytkownik widzi komunikat, że ciśnienie nie powinno mieć wpływu na Jego samopoczucie. W przeciwnym wypadku, gdy ciśnienie wynosi poniżej 1000 hPa lub jest tyle równe, Użytkownik widzi komunikat, że Jego samopoczucie może ulec pogorszeniu, a także inne objawy, jakie mogą wystąpić.[14] Na Rys. 4.21. znajduje się kod pliku barometr.xaml, który jest odpowiada za wygląd. Natomiast Rys. 4.22. i 4.23. przedstawiają kod pliku barometr.xaml.cs.

[10]pobranie daty [**pobranie daty**].
[11]pobranie daty-2 [**pobranie daty-2**].
[12]Wpływ ciśnienia na samopoczucie i stan zdrowia - 1 [**WpĆyw ciŻnienia na samopoczucie i stan zdrowia - 1**].
[13]Wpływ ciśnienia na samopoczucie i stan zdrowia - 2 [**WpĆyw ciŻnienia na samopoczucie i stan zdrowia - 2**].
[14]Ustawienie czcionki [**Ustawienie czcionki**].

Na rysunkach Rys. 4.19. i Rys. 4.20. zostały przedstawione wyglądy stron z barometrem.



Rys. 4.19. Barometr - dobre samopoczucie

Rys. 4.20. Barometr - złe samopoczucie

```
<ContentPage.Content>
    <StackLayout>
        <Label  x:Name="BarometrWynik" BackgroundColor="white"/>
        <Button x:Name="cisnienie" Clicked="Cisnienie_Clicked" Text="zmierz cisnienie"
                HorizontalOptions="CenterAndExpand" BackgroundColor="white" />
        <Label  x:Name="GodzinaLabel" BackgroundColor="white"/>
        <Button x:Name="godzina" Clicked="Godzina_Clicked" Text="aktualna godzina"
                HorizontalOptions="CenterAndExpand" BackgroundColor="white" />

        <Label  x:Name="Samopoczucie" BackgroundColor="Orange" FontSize="20"
                HeightRequest="60" HorizontalTextAlignment="Center"/>

        <Label  x:Name="Tresc" BackgroundColor="white"  FontFamily="FuzzyBubbles-Bold.ttf#FuzzyBubbles-Bold" TextColor="black"
                FontSize="17" MinimumHeightRequest="200"  Padding="20,20,20,20"/>

    </StackLayout>
</ContentPage.Content>
```

Rys. 4.21. barometr.xaml - wygląd strony

```
private void Godzina_Clicked(object sender, EventArgs e)
{
    string godzina = DateTime.Now.Hour.ToString();
    string minuty = DateTime.Now.Minute.ToString();
    string sekundy = DateTime.Now.Second.ToString();
    GodzinaLabel.Text = "Aktualnie mamy godzinę: " + godzina + ":" + minuty + ":" + sekundy;
}

Odwołania: 0
private void Cisnienie_Clicked(object sender, EventArgs e)
{
    if (Barometer.IsMonitoring)
        return;
    Barometer.ReadingChanged += Barometer_ReadingChanged;
    Barometer.Start(SensorSpeed.UI);

}
```

Rys. 4.22. akcje przycisków w pliku cs

```
void Barometer_ReadingChanged(object sender, BarometerChangedEventArgs e)
{
    var data = e.Reading;
    // Process Pressure
    Console.WriteLine($"Reading: Pressure: {data.PressureInHectopascals} hectopascals");

    BarometrWynik.Text = "Ciśnienie wynosi: " + e.Reading.PressureInHectopascals.ToString() + "hPa";


    if(data.PressureInHectopascals <= 1000)
    {
        Samopoczucie.TextColor = Color.Red;
        Samopoczucie.Text = "Twoje samopoczucie może ulec zmianie !!!!!";
        Tresc.Text = "Ze względu na to, że ciśnienie wynosi: " + data.PressureInHectopascals.ToString() +
                     ", może to u Ciebie powodować: " + '\n' +
                     "- osłabienie" + '\n' +
                     "- bóle głowy" + '\n' +
                     "- złe samopoczucie" + '\n' +
                     "- bóle kości, mięśni, stawów" + '\n' +
                     "- skoki ciśnienia tętniczego krwi" + '\n' +
                     "- nadmierna senność lub bezsenność" + '\n' +
                     "- spadek energii życiowej" + '\n' +
                     "- niechęć do aktywności fizycznej" + '\n' +
                     "- wahania nastrojów" + '\n' +
                     "- zaburzenia koncentracji" + '\n';
    }
    else
    {
        Samopoczucie.TextColor = Color.Green;
        Samopoczucie.Text = "Ciśnienie nie powinno mieć wpływu na twoje samopoczucie";
        Tresc.Text = "";
    }


}
```

Rys. 4.23. barometr.xaml.cs

Przy okazji tworzenia barometru, dodana została czcionka i wykorzystana w projekcie. Tekst w aplikacji wygląda na znacznie nowocześniejszy i bardziej przejrzysty. Na Rys. 4.24. został ukazany fragment pliku app.xaml.cs, na którym znajduje się ustawienie wcześniej dodanej do aplikacji czcionki.

```
<Application.Resources>
    <ResourceDictionary>
        <Style TargetType="Label">
            <Setter Property="FontFamily" Value="FuzzyBubbles-Bold.ttf#FuzzyBubbles-Bold"/>

        </Style>
    </ResourceDictionary>
```

Rys. 4.24. Dodanie czcionki w app.xaml.cs

5. Testowanie

6. Podręcznik użytkownika

Spis rysunków

2.1. Grafika Koncepcyjna 7
3.1. Wygląd aplikacji 8
3.2. wygląd po implementacji 9
3.3. wygląd po implementacji ustawienia 10
3.4. Twoje postępy 11
3.5. Strona główna - wygląd 12
3.6. Barometr - wygląd 12
3.7. Krokomierz - wygląd 13
3.8. Postępy - wygląd 13
3.9. Ustawienia - wygląd 14
3.10. Implementacja gradientowego przycisku 14
4.1. Kod wyglądu strony głównej aplikacji 15
4.2. Kod w pliku MainPage.xaml.cs 16
4.3. AndroidManifest.xaml 16
4.4. MainActivity.cs 17
4.5. Ustawienia.xaml.cs 17
4.6. lokalizacja.xaml 18
4.7. lokalizacja 18
4.8. ustawienia.xaml 19
4.9. Ustawienia 19
4.10. Pobieranie wartości przez akcelerometr 20
4.11. Implementacja krokomierza 20
4.12. Krokomierz 21
4.13. Model bazy danych 21
4.14. Połączenie z bazą danych 22
4.15. Postępy.xaml 22
4.16. Krokomierz.xaml.cs 23
4.17. Krokomierz 23
4.18. Twoje postępy 24
4.19. Barometr - dobre samopoczucie 25
4.20. Barometr - złe samopoczucie 25

4.21. barometr.xaml - wygląd strony 26
4.22. akcje przycisków w pliku cs 26
4.23. barometr.xaml.cs 27
4.24. Dodanie czcionki w app.xaml.cs 27

Spis tabel